UNIVERSITÉ DE NANTES

More Automation in Model Driven Development

MEDI 2021

Pascal André, Mohammed El Amin Tebib

LS2N, University of Nantes, France

Davidson, Paris , France

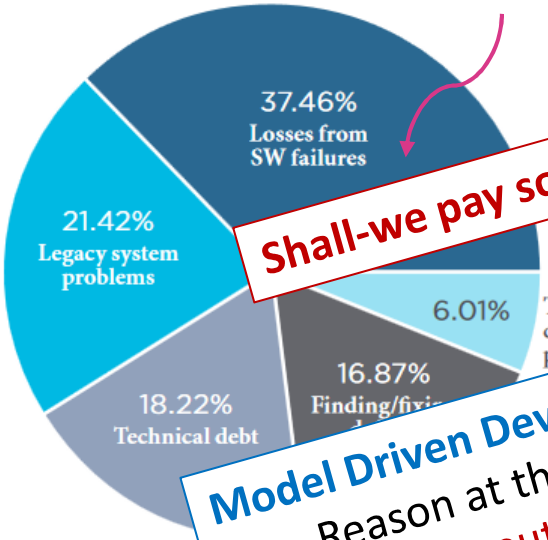
Introduction

software development cost

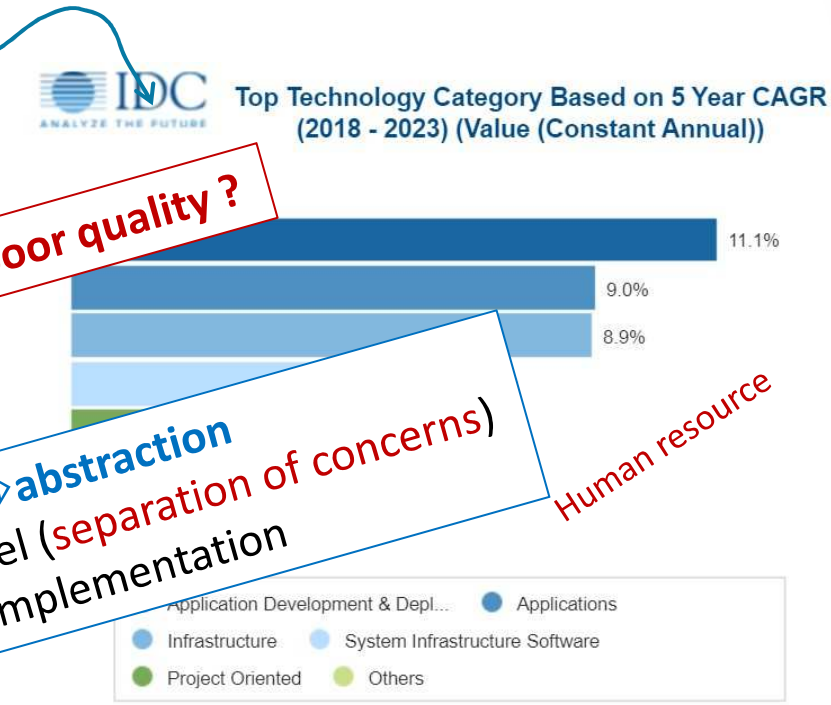
key values findings

- Expensive IT
- Expensive software development
- Expensive low quality

Worldwide IT spending is projected to total $4.1 trillion in 2021, an increase of 8.4% from 2020, according to the latest forecast by Gartner, Inc.



Shall-we pay so much for poor quality ?



Source: IDC Worldwide ICT Spending Guide Industry and Company Size 2019H1

https://www.idc.com/getdoc.jsp?containerId=prUS46047320

Model Driven Development ⇨abstraction

Reason at the model level (separation of concerns)

Partially automate the implementation

the cost of poor … in the US in 2018 is approximately $2 … trillion,

https://www.gartner.com/en/newsroom/press-releases/2021-04-07-gartner-forecasts-worldwide-it-spending-to-reach-4-trillion-in-2021

https://www.it-cisq.org/the-cost-of-poor-quality-software-in-the-us-a-2018-report/The-Cost-of-Poor-Quality-Software-in-the-US-2018-Report.pdf

Introduction

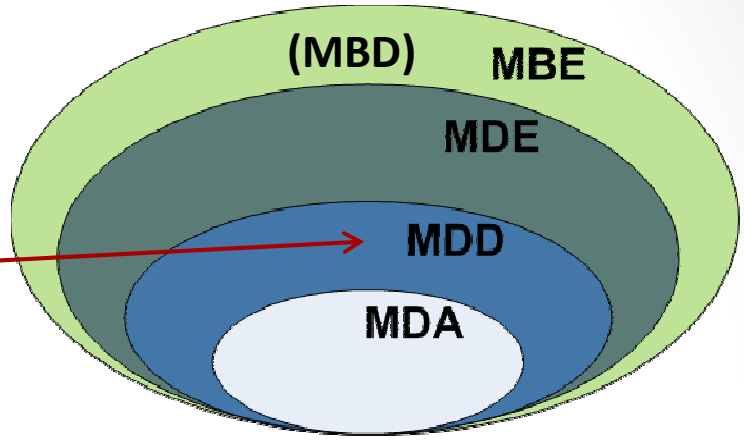
Model Driven Development

- **Context = models**

  MDD

  Models + transformations

  ➥ *Software*



MDSE [Cabot 2012]

- **Problem**

  Model refinement miss automation (see step1)

- **Contribution**

☞ rationalise the process by a structuring frame

☞ increase the role of platform in MT

Outline

- Introduction

  ☞ Contribute to rationalize the development process by a structuring frame

- **Step 1 MDD in practice**
- Step 2 A structuring frame
- Step 3 Implementation and validation
- Conclusion

Step 1 MDD in practice

Three approaches to refine models to code

Student projects

1. Standard implementation
   design and programming

manual

2. Code generation (from high level specifications)
   CASE tools

Fully automated

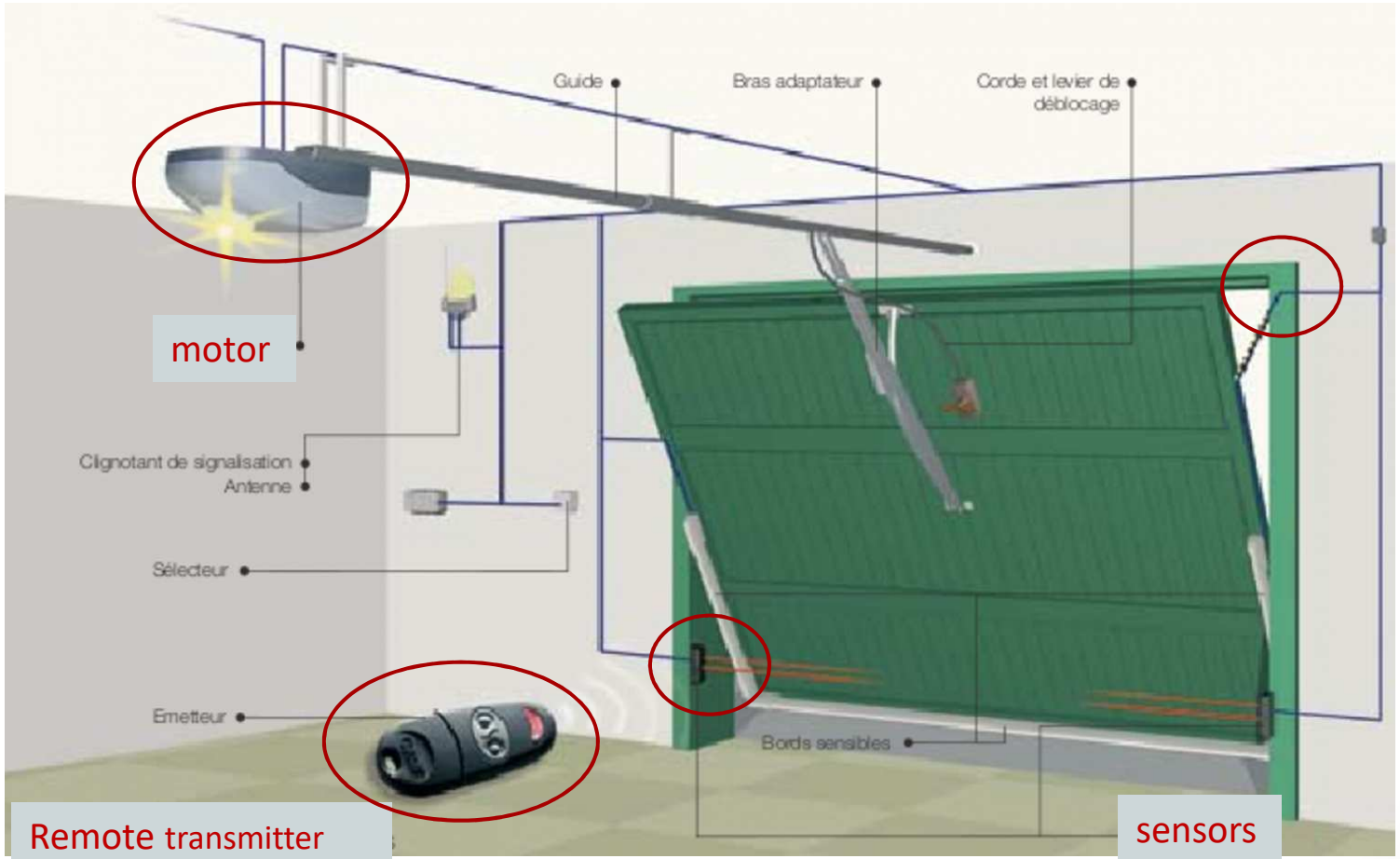
3. Model transformation
   refinement process

Stepwise automation

Experimentations on a representative small case study

Step 1 MDD in practice

Home automation example : Garage door



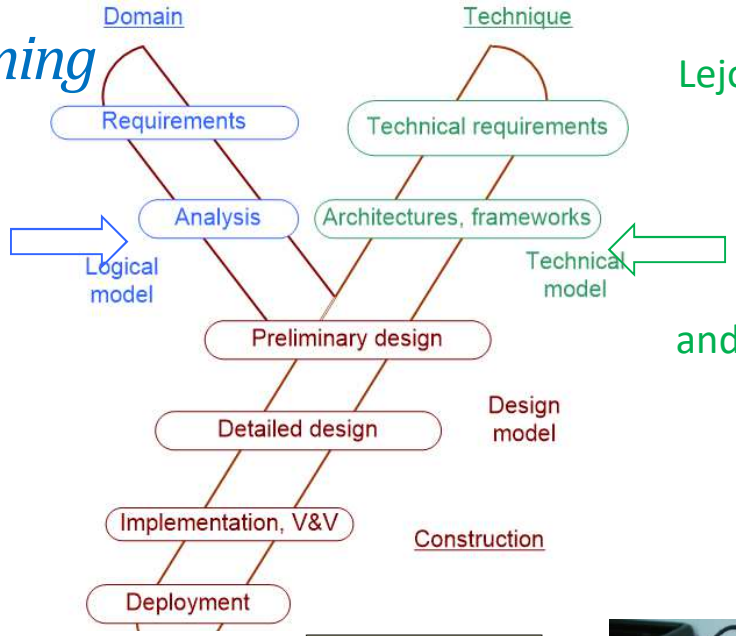
Source: https://www.bricozor.com/automatisme-portes-garages-serie-ver-24-4400-came.html

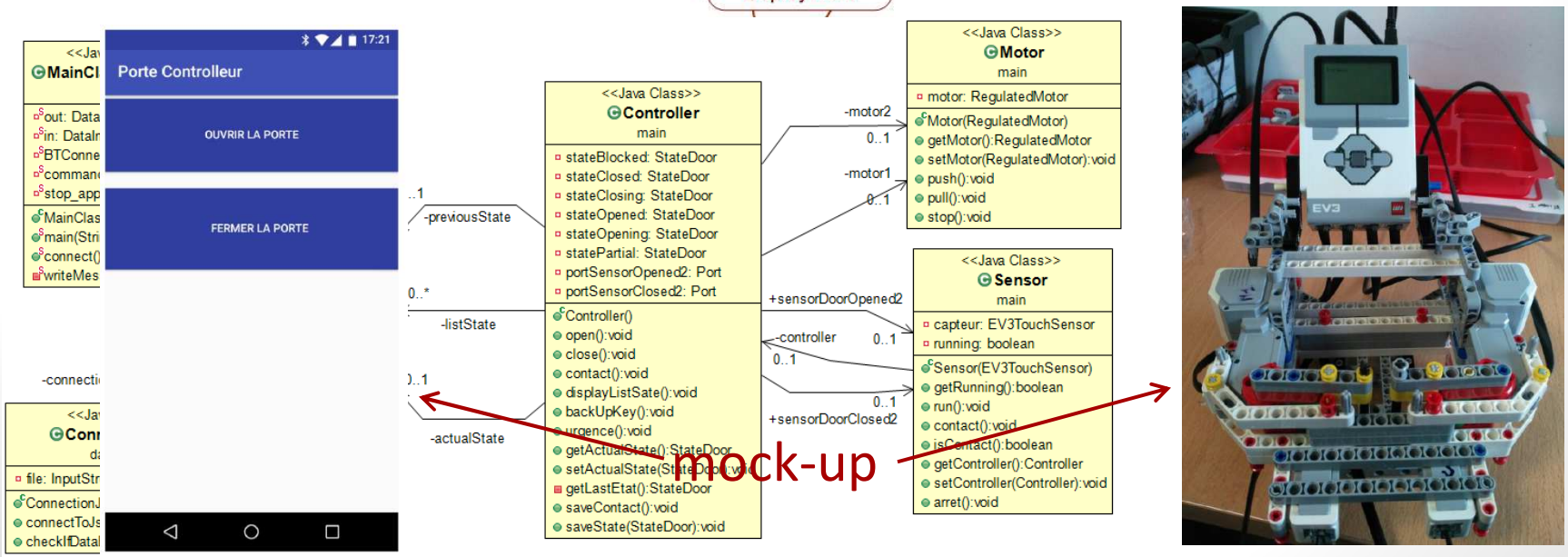
☞*Software Logic model*

Refining models to code

1- design and programming

- Logic model (UML)
- Design / coding
- Testing

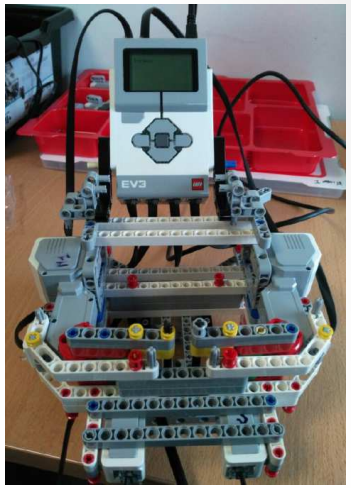




Refining models to code

1- design and programming / feedback

- Operational result = running application
  - The logic model used as a reference not as an abstraction
  - The “functional capability » overrides other software qualities
  - Knowledge and expertise influence the design and programming decision
  - Communication refinement is a first class issue



Partially automating ?

Refining models to code

2- code generation

- Quick UML Case Tools overview
  - Coverage: good for the structure
    low for behaviour (STD) and messages (MOM)
  - Integration : annotations / roundtrip but not API mapping
  - Purpose: simulation (xUML), target code

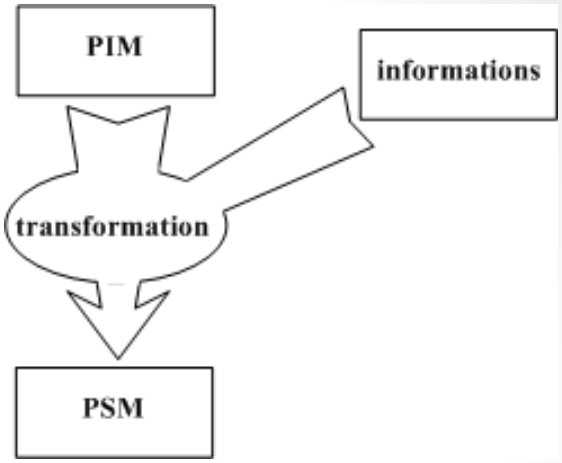
	Star UML	Papyrus	Yakindu	Modelio	VisualParadim	IBM rational rhapsody
UML - XMI	2.0	2.5	-	2.4.1	2.0	2.4.1
CD	√	√	-	√	√	√
STD	-	-	one only	√[1]	√	√
Operations	-	incremental	-	RoundTrip	RoundTrip	√
Round-trip	-	override	-	√	√	√
MOM	-	-	-	-	-	-
API Mapping	-	-	-	-	-	-
Licence[d]	F, C	O	F, C	O	C	C

⇨Stepwise refinement

Refining models to code

3- model transformation

Experimentation feedback



- No generic transformation process
  - Empirical PIM/PSM & Transformations
- Weak Engineering Tool support
  - Languages, Transformations tooling [Kahani2018]
  - Challenging topic [Bucchiarone 2020]
- Most transformations are complex
  - difficult when the source and target model are not semantically closed e.g. *UML statecharts or UML message send – Wifi/BT networks*
  - Insert design decisions in systematic transformation (parameters)
  - Design macro- and micro- transformations
    →Trial and error e.g. tiny ATL transformations (STD)

☞*Key issues for automation*

Step 1 MDD in practice

issues

- Problem of distance between [business] abstractions and [technical] platforms
- Only detailed models can lead to detail code (cf xUML, simulation)
- Design concerns cover cross-cutting concepts
  (Persistence / GUI / distribution / communication)
- No generic transformation process
  - Guidelines
  - Flexibility

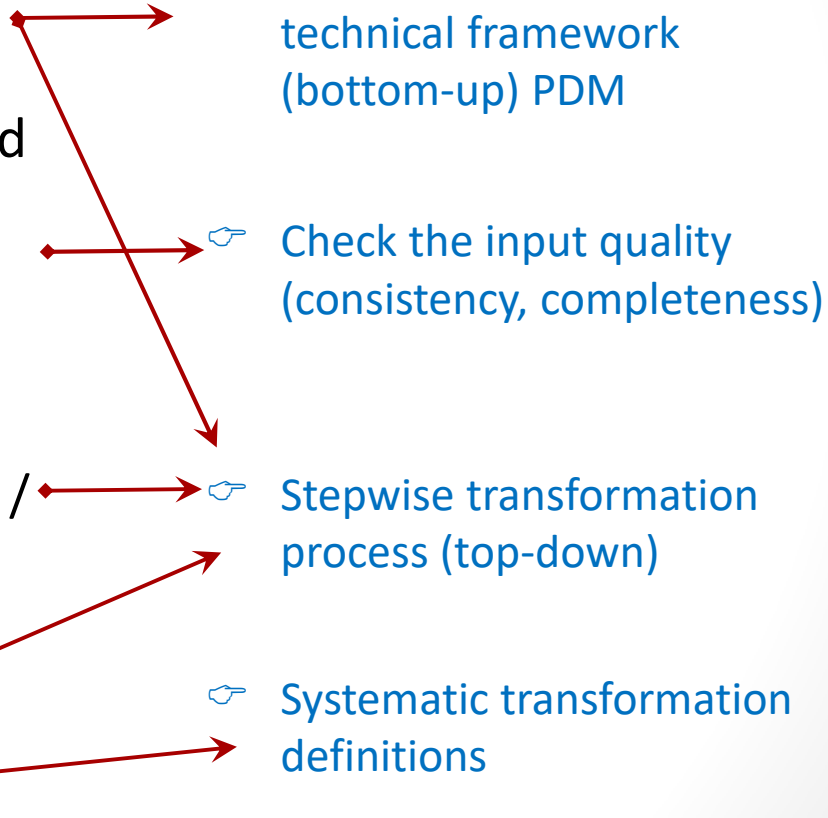
answers ?

Step 2 A structuring frame

Step1 - issues

- Problem of distance between [business] abstractions and [technical] frameworks
- Only detailed models can lead to detail code (cf xUML, simulation)
- Design concerns cover cross-cutting concepts (Persistence / GUI / distribution / communication)
- No generic transformation process
  - Guidelines
  - Flexibility

Step 2 - proposals



- Reverse engineer a technical framework (bottom-up) PDM
- Check the input quality (consistency, completeness)
- Stepwise transformation process (top-down)
- Systematic transformation definitions

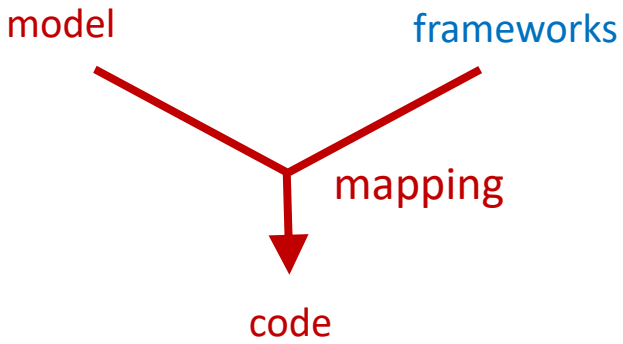
Step 2 A structuring frame

New vision

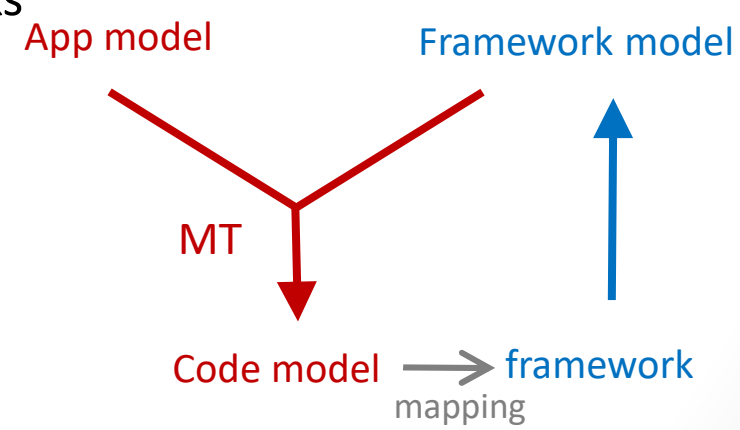
- Refinement AND Mapping

Frameworks

High quality ready to use technical solutions



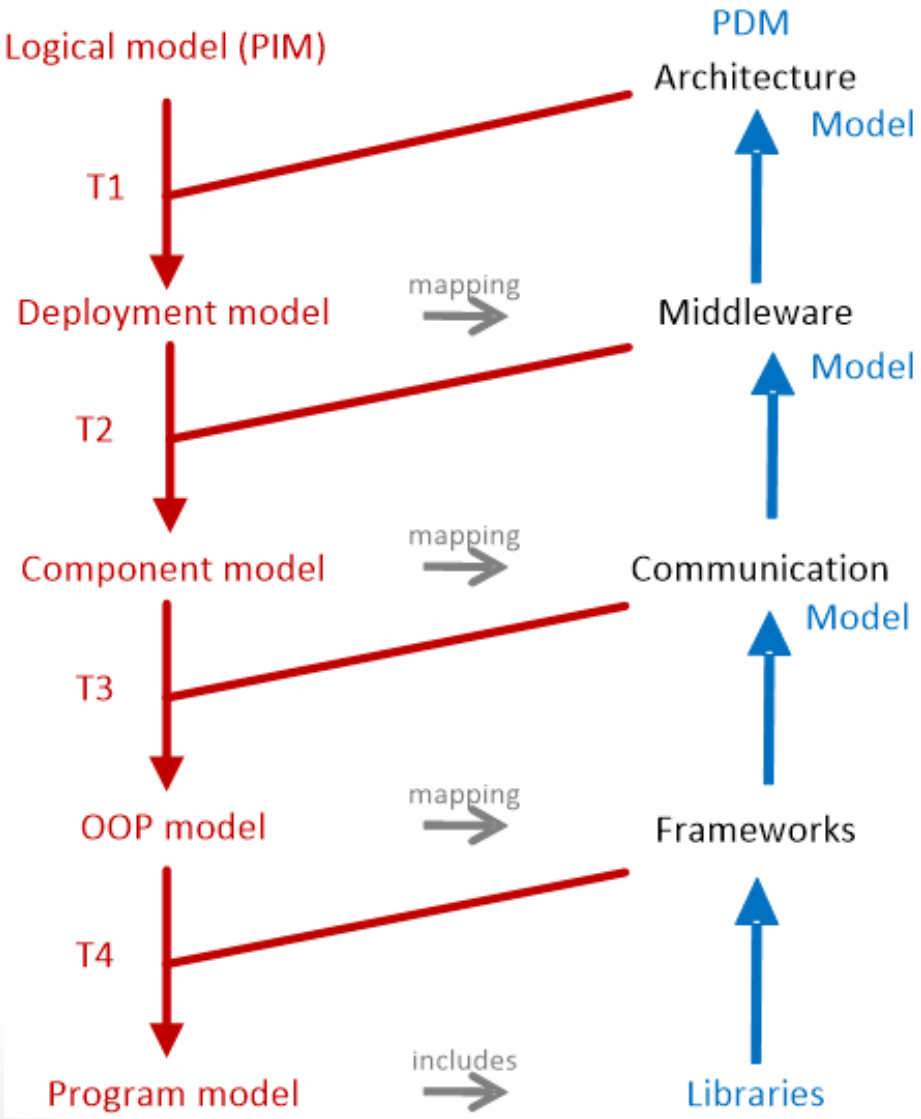
  - Reverse engineer frameworks to enable mapping binding



- Keep the model vision
  - Everything is model (M2M) (except the last M2T to code source)

Step 2 A structuring frame

Step1 - issues



Step 2 - proposals

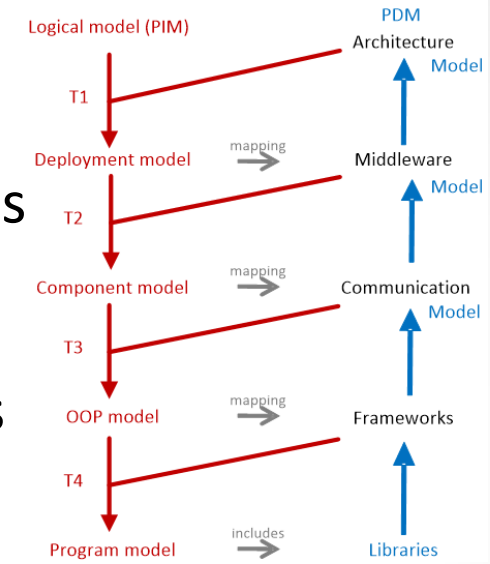
- Reverse engineer a technical framework (bottom-up) PDM
- Check the input quality (consistency, completeness)
- Stepwise transformation process (top-down)
- Systematic transformation definition

Outline

- Introduction
  - ☞ Contribute to rationalize the development process by a structuring frame
- Step 1 MDD practice
- Step 2 Structuring frame
- Step 3 Implementation and validation
  - ☞ First milestones
- Conclusion

Step 3 Implementation and validation

First milestones



- Systematic definitions of transformations
  - T2, T3, T4 (guidelines for T1)
- Model transformation experimentations
  - State-machine transformations (UML/Java profile) - ATL
  - Java code generation - Papyrus
- PDM abstraction and adaptation(T4)
  - Reverse-engineering Lejos (Modisco, AgileJ)
  - API Mapping by Adapter Pattern (ATL)
  - Communication primitives (API)

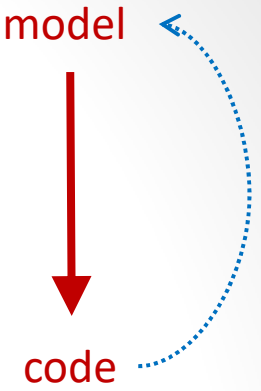
☞ Details in the paper and appendix

Outline

- Introduction
  - ☞ Contribute to rationalize the development process by a structuring frame
- Step 1 MDD practice
- Step 2 Structuring frame
- Step 3 Implementation and validation
- Conclusion

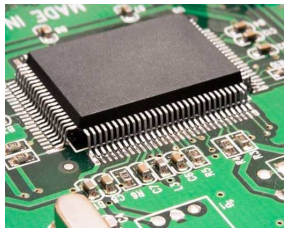
Conclusion

Summary



- Automated MDD is required for “low cost” software development & maintenance
- The problem remains thorny
  - Variety (domains, frameworks, development practice)
  - Tool support not mature enough (Transformation on the shelf)

Proposals

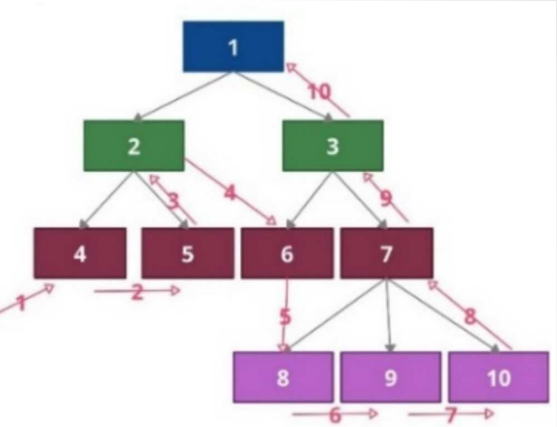


- Generic process of 4 macro-transformations
  - Systematic definition
  - Separation of concerns
- PDM abstraction reverse engineered
- µTransformation implementation (5 %)
  - UML Refinement
  - PDM adaptation

Conclusion

Limits & Open issues

- High-level abstractions and transformations
- Parallelism in the lower level transformation
- Identify the automated/manual parts (GUI)
- Applicability scope & scalability



Ambitious project

Requirements – repositories of

- PDM (framework providers)
- TOST (transformation providers)

Perspectives

- Lejos PDM reverse engineering (contd)
- Communication PDM (contd)
- Full T4 implementation
- Apply a systematic four-step process to the case study

Collaborative contributions are welcome

Thanks for your attention

More Automation in Model Driven Development

A two-track transformation process

https://ev3.univ-nantes.fr/